Supplementary Material
Incorporating domain growth into hybrid methods for reaction-diffusion systems

Cameron A. Smith[1,*], Christian A. Yates[1]
[1]**Centre for Mathematical Biology, Department of Mathematical Sciences, University of Bath, Claverton Down, Bath, BA2 7AY, United Kingdom**

Key words: Reaction–diffusion, domain growth, hybrid methods.

In this document, we introduce any of the mathematics required to demonstrate equivalence between the three methods described in Section 2, and give the numerical algorithms that we employ. It should be noted that other numerical schemes may be used.

S.1 Macroscale

S.1.1 The Lagrangian PDE

We wish to be able to simulate the solution of the PDE (1), with appropriate boundary and initial conditions. However, conventional techniques are unable to do so when the PDE is written in this form due to the existence of the growing domain [Simpson, 2015]. Instead, we transform the PDE onto a fixed coordinate system, known as the Lagrangian coordinates. We apply the following change of variables:

$$x = X\exp\{\rho\tau\}, \tag{S.1}$$

$$t = \tau, \tag{S.2}$$

where $X \in (0, L_0)$ and $\tau > 0$. Applying change of variables (S.1)-(S.2) to the Eulerian PDE (1) yields the Lagrangian PDE

$$\frac{\partial u}{\partial \tau}(X,\tau) = \frac{D}{\exp\{2\rho\tau\}}\frac{\partial^2 u}{\partial X^2}(X,\tau) - \rho u(X,\tau). \tag{S.3}$$

The domain growth manifests itself in equation S.3 in two ways: firstly by creating a time-dependent diffusion coefficient, which comes from the conversion of the second order derivative; and secondly by converting the dilution from an advective term to a particle sink via a first-order degredation reaction. This second term is one of the two terms obtained when the dilution term in (1) is expanded using the product rule. The second of these product rule terms cancels with a term that originates from the conversion of the time derivative.

S.1.2 Numerical scheme

In order to simulate the PDE (S.3), we will use the θ-method (see, for example [Smith, 1985]), however we note that any different simulation methodologies may be appropriate. The θ-method involves writing the second derivative as a combination of implicit and explicit terms, and using a time-stepping algorithm in order to find the solution at a later time. a description can be found in Algorithm S.1 (below) for the PDE (S.3) together with zero-flux boundary conditions. Because the PDE contains time-varying coefficients, the matrices used in order to update the solution need to be calculated at every time-step.

Algorithm S.1: The θ-method (diffusion only)

Input: Current PDE solution vector — $\boldsymbol{U}$; Time of previous update — t_n; Time-step — Δt; θ-value — θ; Diffusion coefficient D; Lagrangian mesh size — h_p; Number of mesh points — $S+1$; Growth rate — ρ; Initial domain length L_0.

(1a) Calculate the time-dependent matrices A_n and B_n:

$$(A_n)_{ij} = \begin{cases} 1 + \frac{D\Delta t\theta}{h_p^2}\exp\{-2\rho(t_n+\Delta t)\} & i = j = 1, S+1 \\ 1 + 2\frac{D\Delta t\theta}{h_p^2}\exp\{-2\rho(t_n+\Delta t)\} & i \in \{2,...,S\}, j = i \\ -\frac{D\Delta t\theta}{h_p^2}\exp\{-2\rho(t_n+\Delta t)\} & i \in \{2,...,S+1\}, j = i-1 \\ -\frac{D\Delta t\theta}{h_p^2}\exp\{-2\rho(t_n+\Delta t)\} & i \in \{1,...,S\}, j = i+1 \\ 0 & \text{otherwise} \end{cases}$$

$$(B_n)_{ij} = \begin{cases} 1 - (\rho+\mu)\Delta t - \frac{D\Delta t(1-\theta)}{h_p^2}\exp\{-2\rho t_n\} & i = j = 1, S+1 \\ 1 - (\rho+\mu)\Delta t - 2\frac{D\Delta t(1-\theta)}{h_p^2}\exp\{-2\rho t_n\} & i \in \{2,...,S\}, j = i \\ -\frac{D\Delta t(1-\theta)}{h_p^2}\exp\{-2\rho t_n\} & i \in \{2,...,S+1\}, j = i-1 \\ -\frac{D\Delta t(1-\theta)}{h_p^2}\exp\{-2\rho t_n\} & i \in \{1,...,S\}, j = i+1 \\ 0 & \text{otherwise} \end{cases}$$

(1b) Update the PDE solution $\boldsymbol{U}$ according to the following:

$$\boldsymbol{U} \leftarrow A_n^{-1} B_n \boldsymbol{U}.$$

S.1.3 Remeshing

Due to the difference in coordinate systems described in Section 3 of the main text, the solution vector is calculated on a static Lagrangian mesh. Considered in Eulerian coordinates, however, this mesh grows. As such, it is prudent to remesh the solution vector when the PDE mesh grows by a certain amount in Eulerian coordinates. We present the remeshing algorithm which requires the pre-calculation of the times that we remesh. A natural time to remesh is when the domain has doubled in length, and this is indeed how we calculate these remeshing times. However, alternative scale factors may be used.

We begin by calculating the number of times the domain doubles in length before the final time of the simulation, t_f, via

$$s_{max} = \lfloor \exp\{\rho t_f\}/2 \rfloor, \tag{S.4}$$

where $\lfloor \cdot \rfloor$ denotes the floor function. Then for $i \in \{1,...,s_{max}\}$, the ith time of remeshing is given by solving

$$2iL_0 = L_0 \exp\{\rho t_i\}. \tag{S.5}$$

The full algorithm for remeshing can be found in Algorithm S.2

Algorithm S.2: Remeshing the PDE

Input: Current PDE solution vector — $\boldsymbol{U}$; Growth rate — ρ; Current number of PDE mesh points — $S+1$; Current remesh time — t_i; Next remesh time — t_{i+1}.

(2a) Create a temporary solution vector $\boldsymbol{V}$ of size $2S+1$.

(2b) Set $V_{2j-1} = U_j$ for every $j \in \{1, ..., S+1\}$.

(2c) For all V_j with $j \in \{2, 4, 6, ..., 2S\}$, interpolate the solution. Any reasonable interpolation may be used, however for the purposes of this paper we use linear interpolation:

$$V_j = \frac{V_{j-1} + V_{j+1}}{2}, \quad j \in \{2, 4, 6, ..., 2S\}$$

(2d) Set $\boldsymbol{U} \leftarrow \boldsymbol{V}$.

(2e) Update the number of mesh points to be $2S+1$.

(2f) Update the remesh time to be t_{i+1}.

S.2 Mesoscale

S.2.1 Modified next reaction method

We will evolve the mesoscopic system according to the modified next reaction method [Anderson, 2007]. We choose to use the modified next reaction method as opposed to the standard Gillespie algorithm [Gillespie, 1977] because the propensity functions in the hybrid methods in the main text depend explicitly on time, meaning that between two events taking place, the propensity function is changing. This violates the assumption of the Gillespie algorithm, that requires the propensity functions to remain constant between the times that events take place.

In order to use the modified next reaction method, we require the definition of propensity functions $a_j(\boldsymbol{n}, t)$, where $j \in \{1, ..., 2K(t)\}$ indexes the possible events that can occur, with the first $2K(t)$ being the diffusive jumps left and right from each compartment. The probability of a particular event j occurring during a small time interval $(t, t+\delta t)$ is then defined to be $a_j(\boldsymbol{n}(t), t)\delta t$.

The modified next reaction method requires the calculation and tracking of internal clock times T_i for every possible event, and next firing times P_i. These are initialised as $T_i = 0$ and $P_i = \ln(1/r_i)$, where r_i is a uniformly distributed random variable between zero and one. Propensity functions a_i are initialised, and the absolute time until the next event of type i fires can be calculated by solving:

$$P_i - T_i = \int_t^{t+\Delta t_i} a_i(\boldsymbol{n}(t), s)\ ds, \tag{S.6}$$

for Δt_i at $t = 0$.

The algorithm then locates the event which happens first, which is equivalent to finding the minimum of the Δ_i, which we call Δ. This event is enacted, all internal clock times are updated according to

$$T_i \leftarrow T_i + \int_t^{t+\Delta} a_i(\boldsymbol{n}(t), s)\ ds, \tag{S.7}$$

and for the event that fires, say event β, a new next firing time is found by setting

$$P_\beta \leftarrow P_\beta + \ln\left(\frac{1}{r_\beta}\right), \tag{S.8}$$

where r_β is a uniformly distributed random variable between zero and one. Time is updated to be $t + \Delta$ and then the algorithm repeats. The time-varying nature of the propensity functions is accounted for by using the internal clocks as opposed to absolute time. The algorithm for the modified next reaction method can be found in Algorithm S.3.

Algorithm S.3: Modified next reaction method [Anderson, 2007] (diffusion only)

Input: Current mesoscopic state — $\boldsymbol{n}$, Current time — t; Final time — t_f; Current number of compartments — K; Internal clock times — T_i for $i \in \{1, ..., 2K\}$; Next firing times — P_i for $i \in \{1, ..., 2K\}$; Propensity function — a_i for $i \in \{1, ..., 2K\}$; Times until next event — Δt_i for $i \in \{1, ..., 2K\}$ (see equation (S.6)).

(3a) Find $\Delta = \min\{i \in \{1, ..., 2K\} : \Delta t_i\}$ and $\beta = \text{argmin}\{i \in \{1, ..., 2K\} : \Delta t_i\}$.

(3b) Enact the event β.

(3c) Update the internal clocks according to equation (S.7).

(3d) For event β, update the next firing time using equation (S.8).

(3e) Recalculate the propensity functions at the new time.

(3f) Update absolute time $t = t + \Delta$.

(3g) If $t < t_f$, return to step **(S.3a)**, otherwise end.

S.2.2 Domain stretching

Next we address how to stretch the domain, which we do according to the stretching method [Smith et al., 2019]. Suppose we have a state $\boldsymbol{N}^{K-1}(t)$ which contains $K(t) - 1$ compartments, and we wish to extend the domain by a single compartment. In order to do this, we will define the pre-growth state $\boldsymbol{N}^{K-1}(t) = \boldsymbol{r} \in \mathbb{R}^{K(t)-1}$ and the post-growth state to be $\boldsymbol{N}^{K}(t)\boldsymbol{n} \in \mathbb{R}^{K(t)}$. In order to determine how the particles in the pre-growth state should be redistributed to the post-growth state, we will calculate "overlap regions". Consider the compartments being of length h_c. When there are $K(t)$ compartments, we have a domain of length $K(t)h_c$. We now stretch our pre-growth state to be of the same length, meaning that each of the $K(t) - 1$ compartments are of length $h_c K(t)/(K(t) - 1)$, yielding a domain of length $K(t)h_c$. From these, we can calculate the length of the i^{th} pre-growth compartment that overlaps the i^{th} post-growth compartment. These are the overlap regions. Using this set up, it can be calculated that the i^{th} overlap region, with $K(t) - 1$ pre-growth compartments, is of size

$$\delta_i^{K(t)-1} = \frac{K(t) - i}{K(t)}, \tag{S.9}$$

which holds for $i \in \{1, ..., K(t) - 1\}$.

These overlap regions are treated as the probability of placing a particle from pre-growth compartment i, into post-growth compartment i, independent of all others. Therefore, for each pre-growth compartment, draw a Binomially distributed random variable b_i with r_i trials and probability of success $\delta_i^{K(t)-1}$.

Then for each post-growth compartment $j \in \{1, ..., K(t)\}$ we set:

$$n_j = \begin{cases} b_1, & \text{if } j = 1, \\ b_j + (r_{j-1} - b_{j-1}), & \text{if } j \in \{2, ..., K(t) - 1\}, \\ r_{K(t)-1} - b_{K(t)-1}, & \text{if } j = K(t). \end{cases} \tag{S.10}$$

We then set the new current state $\boldsymbol{N}(t)$ to be $\boldsymbol{n}$. This algorithm can be found in Algorithm S.4.

Algorithm S.4: The stretching method [Smith et al., 2019]

Input: Current mesoscopic state — $\boldsymbol{n}$; Current number of compartmemts — K.

(4a) Define the pregrowth state to be $\boldsymbol{r} \leftarrow \boldsymbol{n}$.

(4b) Calculate the overlap proportions $\delta_i^K = (K + 1 - i)/(K + 1)$.

(4c) Draw K binomial random variables $b_i \sim \text{Bin}(r_i, \delta_i^K)$.

(4d) Create an extra compartment at the right end of the postgrowth domain (increasing K by 1 by setting $K \leftarrow K + 1$).

(4e) For $j \in \{1, ..., K\}$, set n_j according to (S.10).

We can incorporate the stretching into the overall mesoscopic simulation algorithm in two ways. The first is to add an extra propensity function to the list which represents the stretching event. Over multiple repetitions of the algorithm, this will yield domains of different lengths due to the stochasticity in the number of times this event will fire. However, in the limit as the number of repeats tends to infinity, the length of the domain will become the average length of the domain, which will be the length of the equivalent PDE domain, i.e. $L(t) = L_0 \exp\{\rho t\}$. We will not take this approach, as we wish to assess the accuracy of the hybrid method, which is aided by the removal of as many sources of statistical error from other parts of the algorithm as possible. Alternatively, we use the deterministic length $L(t) = L_0 \exp\{\rho t\}$, to calculate the times at which, on average, the number of compartments should increase by 1, and will always enact a stretching event at these times.

S.2.3 Equivalence

To demonstrate equivalence of this mesoscopic scheme to the Eulerian PDE, we need to formulate the equations for the evolution of the mean number of particles in the mesoscopic description, which we do by firstly writing down the master equation for the system. Define $p(\boldsymbol{n}, k, t)$ to be the probability that the state variable $\boldsymbol{N}^K(t)$ is $\boldsymbol{n}$ and the number of compartments $K(t)$ is k at time t. The master equation describes the evolution of this probability. Then the rate of change of this probability is

$$\begin{aligned} \frac{\partial p}{\partial t}(\boldsymbol{n}, k, t) = d \sum_{i=1}^{k-1} \left[(n_i + 1) p(J_i^+ \boldsymbol{n}, k, t) - n_i p(\boldsymbol{n}, k, t) \right] + d \sum_{i=2}^{k} \left[(n_i + 1) p(J_i^- \boldsymbol{n}, k, t) - n_i p(\boldsymbol{n}, k, t) \right] \\ + \rho(k-1) \sum_{\boldsymbol{r} \in \mathcal{M}_{k-1}^M} \left[p(\boldsymbol{r}, k-1, t) \pi(\boldsymbol{n}|\boldsymbol{r}) \right] - \rho k p(\boldsymbol{n}, k, t). \end{aligned} \tag{S.11}$$

Here, we have that d is the rate for any particle to jump from its compartment to one of its neighbours, $J_i^{\pm}$ are operators which move a single particle from compartment i to compartment $i \pm 1$, then set $\mathcal{M}_k^M = \left\{ \boldsymbol{m} \in \mathbb{N}^k : \sum_{i=1}^k m_i = M \right\}$ is the set of all state variables with k compartments and M total particles, and $\pi(\boldsymbol{n}|\boldsymbol{r})$ is the transition probability from state $\boldsymbol{N}^{k-1} = \boldsymbol{r}$ to $\boldsymbol{N}^k = \boldsymbol{n}$. The first two sums on the right-hand side capture the impact of the diffusive jumps right and left respectively, and the final two terms correspond to the effects of domain growth.

In order to obtain the mean equations, we define

$$\bar{M}_i^k(t) = \sum_{\boldsymbol{n} \in \mathcal{M}_k^M} n_i p(\boldsymbol{n}, k, t). \tag{S.12}$$

to be the average number of particles in compartment i when there are k compartments in total, at time t. We can then calculate the mean equations by multiplying equation (S.11) by n_i and summing over the state space $\boldsymbol{n}$ to give:

$$\frac{\mathrm{d}\bar{M}_i^k}{\mathrm{d}t} = d\bar{M}_{i-1}^k - (2d + \rho k)\bar{M}_i^k + d\bar{M}_{i+1}^k + \rho(k-1)\left[\left(1 - \frac{k-i+1}{k}\right)\bar{M}_{i-1}^{k-1} + \frac{k-i}{k}\bar{M}_i^{k-1}\right]. \tag{S.13}$$

In order to write this as a continuous system, we let $x = ih_c$, and suppose that $\bar{M}_i^k(t) \approx u(x,t)$ and $\bar{M}_{i\pm1}^k(t) \approx u(x \pm h_c, t)$. We will finally assume that mass spreads uniformly when it grows, so that $(k-1)\bar{M}_i^{k-1} = k\bar{M}_i^k$. Substituting this into the mean equations (S.13) yields

$$\begin{aligned}\frac{\partial u}{\partial t}(x,t) \approx{}& du(x-h_c,t) - 2du(x,t) + du(x+h_c,t)) \\ &+ \rho\frac{k(i-1)}{k}u(x-h_c,t) + \rho\frac{k(k-i)}{k}u(x,t) - \rho k u(x,t).\end{aligned}$$

We apply a second order Taylor expansion about x and rearrange in derivatives of u. We also omit all arguments of the function u as all will be evaluated at (x,t). This gives us the following:

$$\begin{aligned}\frac{\partial u}{\partial t} &\approx d\left[u - h_c\frac{\partial u}{\partial x} + \frac{h_c^2}{2}\frac{\partial^2 u}{\partial x^2}\right] - 2du + d\left[u + h_c\frac{\partial u}{\partial x} + \frac{h_c^2}{2}\frac{\partial^2 u}{\partial x^2}\right] \\ &\qquad + \rho(i-1)\left[u - h_c\frac{\partial u}{\partial x} + \frac{h_c^2}{2}\frac{\partial^2 u}{\partial x^2}\right] + \rho(k-i)i - \rho k u, \\ &= \left[d - 2d + d + \rho(i-1) + \rho(k-i) - \rho k\right]u + h_c\left[-d + d - \rho(i-1)\right]\frac{\partial u}{\partial x} \\ &\qquad + \frac{h_c^2}{2}\left[d + d + \rho(i-1)\right]\frac{\partial^2 u}{\partial x^2}, \\ &= -\rho u - \rho h_c(i-1)\frac{\partial u}{\partial x} + dh_c^2\frac{\partial^2 u}{\partial x^2} + \frac{\rho}{2}h_c^2(i-1)\frac{\partial^2 u}{\partial x^2}.\end{aligned}$$

We utilise the fact that $ih_c = x$ in order to simplify this further:

$$\frac{\partial u}{\partial t} = -\rho u - \rho x\frac{\partial u}{\partial x} + dh_c^2\frac{\partial^2 u}{\partial x^2} + h_c\left[\rho\frac{\partial u}{\partial x} + \frac{\rho}{2}(x - h_c)\frac{\partial^2 u}{\partial x^2}\right].$$

We now take the diffusive limit, by taking compartment size, h_c to 0 while making the jump rate, d, infinite and keeping dh_c^2 constant. We also recognise the first two terms as being an expansion of the derivative of $-\rho u x$ to obtain

$$\frac{\partial u}{\partial t} = -\rho\frac{\partial(ux)}{\partial x} + dh_c^2\frac{\partial^2 u}{\partial x^2}. \tag{S.14}$$

Comparing this with equation (1), we note that equivalence requires the Fickian diffusion coefficient D to be related to the mesoscopic jump rate d via $D = dh_c^2$.

S.3 Microscale

S.3.1 Numerical scheme

In order to simulate the SDE (4), we utilise the Euler-Maruyama method. Suppose we have a system containing N particles, whose positions at time t are given by $y_i(t)$. Then the Euler-Maruyama method allows us to update the positions in the time interval $(t, t+\Delta t)$ according to

$$y_i(t+\Delta t) = y_i(t) + \rho y_i(t)\Delta t + \sqrt{2D\Delta t}\ \xi_i, \tag{S.15}$$

where the ξ_i are normally distributed random variables with zero mean and unit variance. The algorithm for the movement of particles can be found in Algorithm S.5

Algorithm S.5: Individual-based update (diffusion only)

Input: Current positions of particles — $\boldsymbol{y}$; Current time — t; Time-step to evolve particles — Δt; Diffusion coefficient — D; Growth rate — ρ;

(5a) Update the particles' positions according to the computational SDE (S.15).

(5b) For every y_i such that $y_i < X_0 \exp\{\rho(t+\Delta t)\}$, set

$$y_i \leftarrow y_i + 2(X_0 \exp\{\rho(t+\Delta t)\} - y_i).$$

(5c) For every $y_i > X_1 \exp\{\rho(t+\Delta t)\}$, set

$$y_i \leftarrow y_i - 2(y_i - X1 \exp\{\rho(t+\Delta t)\}).$$

We now briefly discuss how reactions are to be enacted. Zeroth-order reactions, where particles appear without the influence of any other particle, will occur with a probability $\kappa_0\Delta t$ over a time interval $(t, t+\Delta t)$, where κ_0 is the rate per unit time of the reaction. First-order reactions depend on the particles already present in the system. For every particle, the reaction is enacted with a probability $\kappa_1\Delta t$, where again, the κ_1 is the rate for each particle to react. Second-order reactions involve the interaction of two particles that are "close enough" to react. There are several methods that are able to resolve these interactions, such as the λ-ρ method [Erban and Chapman, 2009]. We refer to those interested in such methods to the following references [Smoluchowski, 1917, Andrews and Bray, 2004, Erban and Chapman, 2009, Lipková et al., 2011, van Zon and ten Wolde, 2005].

S.3.2 Equivalence

Finally, to establish equivalence with the macroscale, we define the probability of finding a particle located at a position x at time t, given that it was at a position y at time $s < t$ to be $q(x, t|y, s)$. Then, through the use of an intermediate point, we can write the Chapman-Kolmogorov equation

$$q(z, t+\Delta t|y, s) = \int_{-\infty}^{\infty} q(z, t+\Delta t|x, t) q(x, t|y, s)\ dx.$$

This equation holds for any Δt, not necessarily small. We can multiply this equation by a smooth test function $\varphi(z)$, integrate over z and relabel the integral on the LHS to be an integral with respect to x

instead of z to yield

$$\int_{-\infty}^{\infty} q(x, t+\Delta t|y, s)\varphi(x)\ dx = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} q(z, t+\Delta t|x, t)\varphi(z)\ dz\right] q(x, t|y, s)\ dx. \tag{S.16}$$

Taylor expanding $\varphi(z)$ on the RHS about x to second order, we obtain three integrals which can be interpreted as the zeroth- first- and second-order moments of the distribution $q(\cdot, t+\Delta t|z, t)$. These quantities are found straightforwardly using the SDE (4). Once substituted into equation (S.16), all derivatives on the φ function are transferred to the q function, and a rearrangement yields the Eulerian PDE (1), known in this context as the Fokker-Planck equation.

S.4 Additional plots

In this section, we present any additional plots to supplement the test problems. In Figures S.1–S.3, we present the difference in particle numbers between the hybrid methods and ground truth in each subdomain for each of the test problems in Section 6.

References

D.F. Anderson. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *J. Chem. Phys.*, 127(21):214107, 2007.

S.A. Andrews and D. Bray. Stochastic simulation of chemical reactions with spatial resolution and single molecule detail. *Phys. Biol.*, 1(3-4):137–151, 2004.

R. Erban and S.J. Chapman. Stochastic modelling of reaction–diffusion processes: algorithms for bimolecular reactions. *Phys. Biol.*, 6(4):1–18, 2009.

D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25): 2340–2361, 1977.

J. Lipková, K.C. Zygalakis, S.J. Chapman, and R. Erban. Analysis of Brownian dynamics simulations of reversible bimolecular reactions. *SIAM J. Appl. Math.*, 71(3):714–730, 2011.

M.J. Simpson. Exact solutions of linear reaction-diffusion processes on a uniformly growing domain: Criteria for successful colonization. *PLoS One*, 10(2):e0117949, 2015.

C.A. Smith, C Mailler, and C.A Yates. Unbiased on-lattice domain growth. *Phys. Rev. E*, 100(6), dec 2019.

G.D. Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford University Press, 1985.

M. Smoluchowski. Versuch einer mathematischen theorie der koagulationskinetik kolloider lösungen. *Z. Phys. Chem.*, 92(129-168):9, 1917.

J.S. van Zon and P.R. ten Wolde. Green's-function reaction dynamics: a particle-based approach for simulating biochemical networks in time and space. *J. Chem. Phys.*, 123(23):234910, 2005.

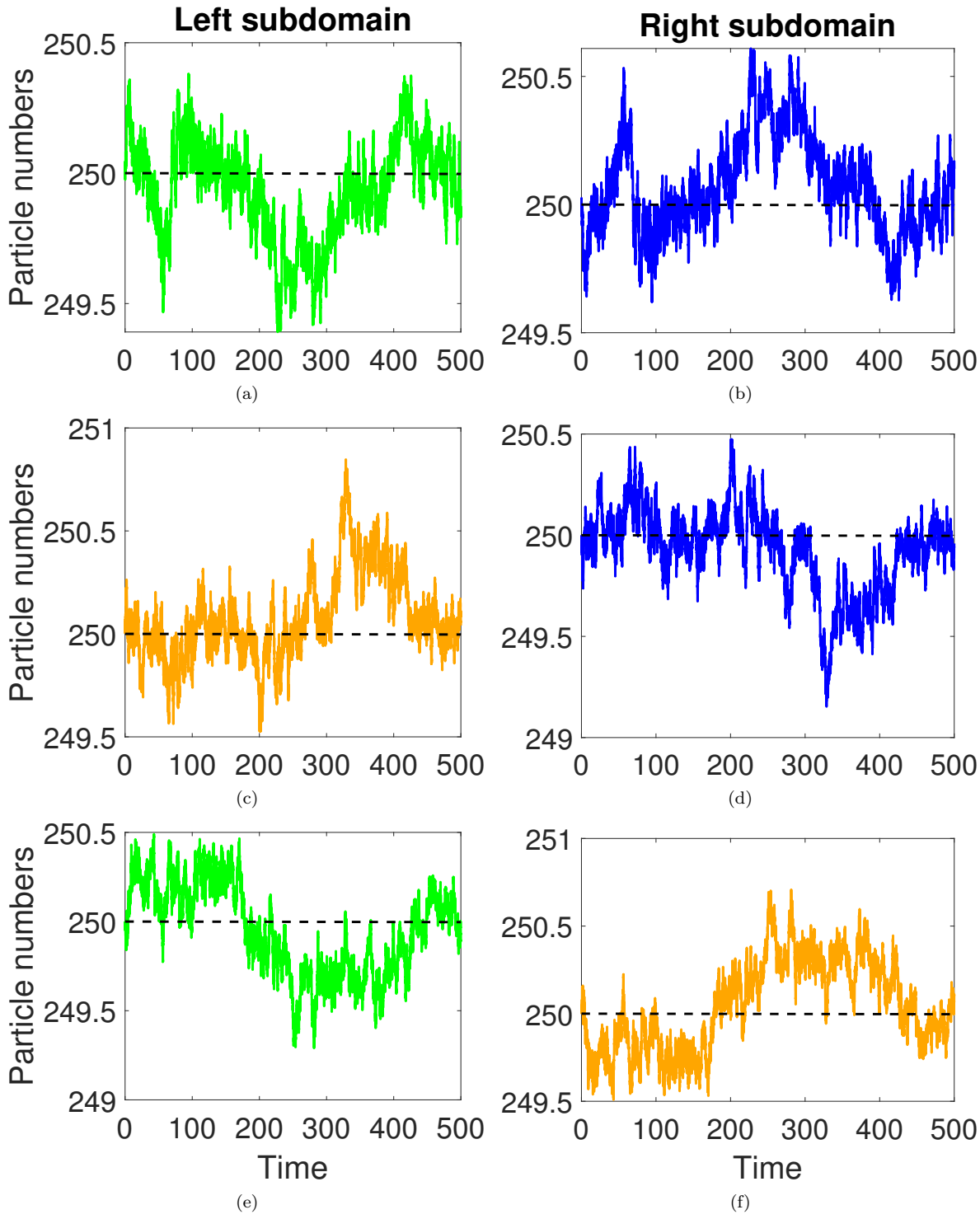

Figure S.1. The difference between the numbers of particles in each subdomain for test problem 1. The top row are the numbers are for the PCM, the second row is the GCM, and the bottom row is the ARM. The left column denotes the particle numbers in the left subdomin, while the right column is in the right subdomain. The coloured lines are the numbers of particles in the subdomain if it is represented by the PDE (green), compartments (blue) and Brownian-based (orange). The black dashed line is the number of particles as calculated by the numerical solution of the PDE over the entire domain.

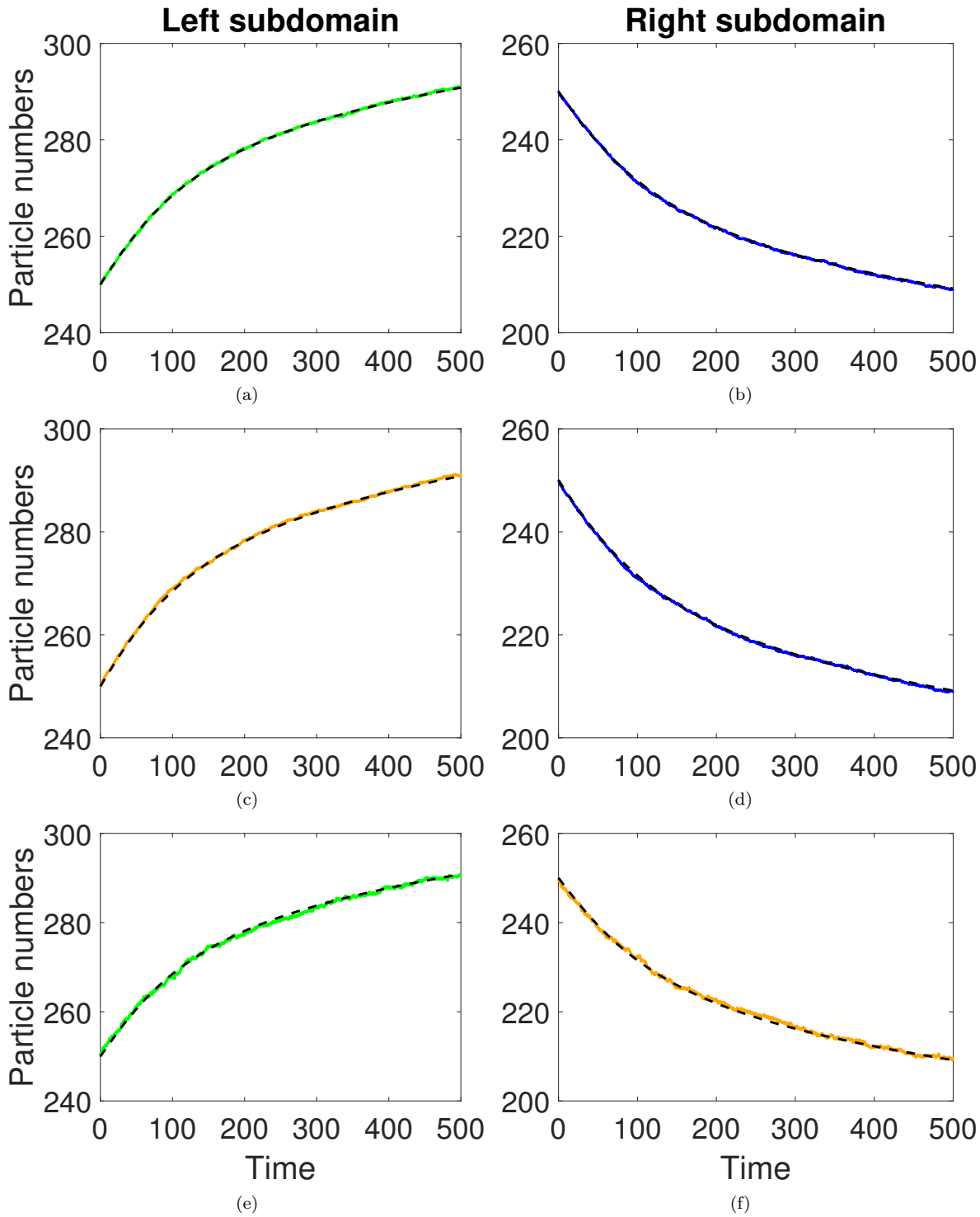

Figure S.2. The difference between the numbers of particles in each subdomain for test problem 2. All colours and labels are the same as for Figure S.1

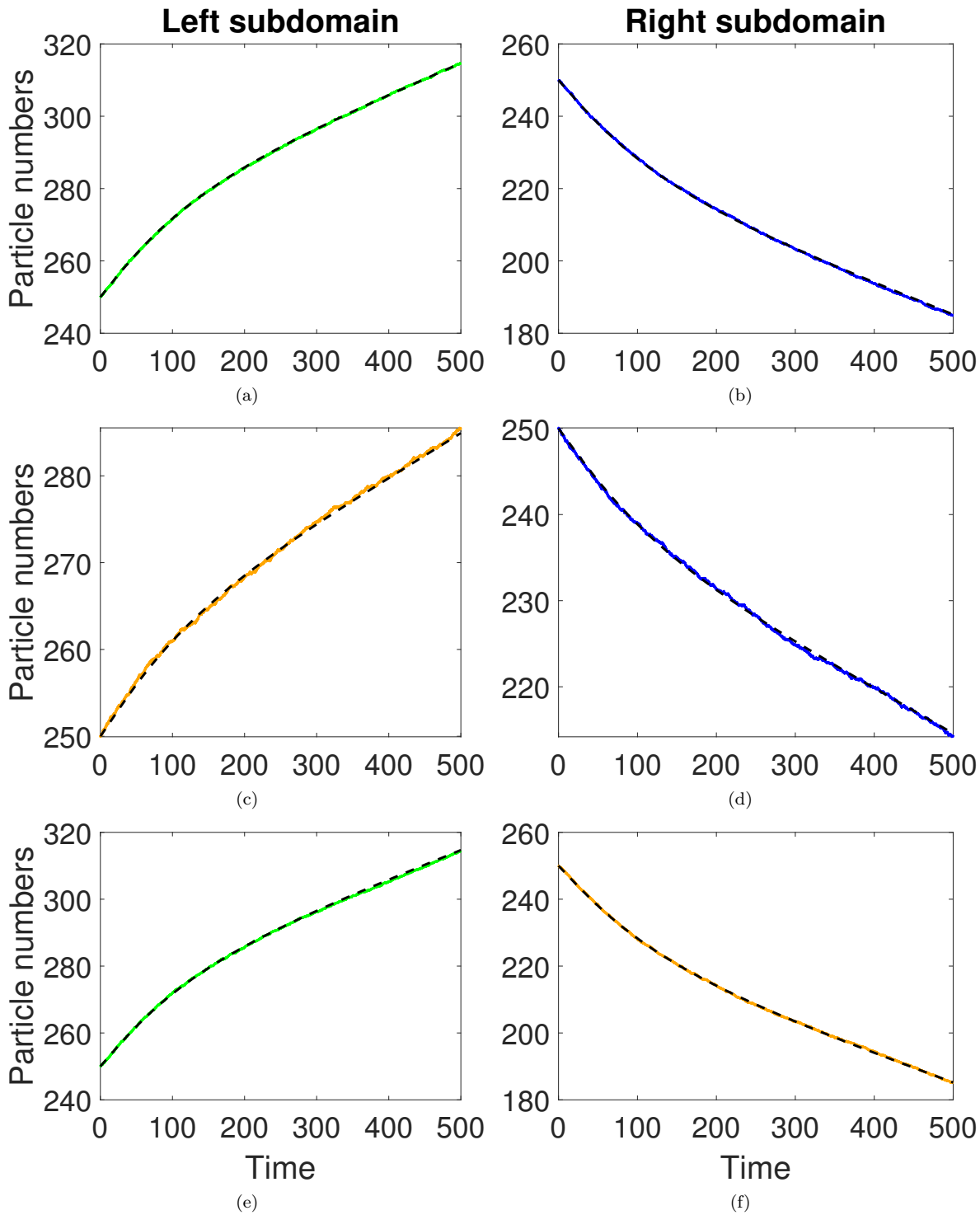

Figure S.3. The difference between the numbers of particles in each subdomain for test problem 3. All colours and labels are the same as for Figure S.1